UNITED STATES DISTRICT COURT
DISTRICT OF MASSACHUSETTS

FRED B. DUFRESNE, Plaintiff, v. MICROSOFT CORPORATION, ADOBE SYSTEMS INCORPORATED and MACROMEDIA, INCORPORATED, Defendants.	Civil Action No. 02-11778 WGY

DEFENDANTS' OPENING CLAIM CONSTRUCTION BRIEF

TABLE OF CONTENTS

Page

I. Introduction 1

II. Background 1

A. The Internet 1

B. Webpages And The Hypertext Markup Language (HTML) 1

C. Dynamic Versus Static Webpages 2

D. The Prosecution History 6

III. Canons Of Claim Construction 9

IV. Argument 10

A. "Executable Tag" (All Claims) 10

1. Defendants' Proposed Construction Is Compelled By The Intrinsic And Extrinsic Evidence 10

2. Plaintiff's Proposed Construction Flouts The Intrinsic Record 12

B. "Hypertext Source To A Displayable Page" (All Claims) 13

1. Defendants' Proposed Construction Is Dictated By The Intrinsic Record 13

2. Plaintiff's Proposed Construction 14

C. "Processing The Source … By Executing The Tags"/"Processing The Source …, The Process Executing The Executable Tags" (All Claims) 14

1. Defendants' Proposed Construction 14

2. Plaintiff's Proposed Construction 15

D. "Server" (All Claims) 16

E. "Client" (All Claims) 16

F. "Hypertext Markup Codes"/"Markup Codes" (Claims 9 and 26) 17

G. "Embedded Within The [Markup] Codes" (Claims 9 and 26) 18

H. "Database" (Claim 26) 18

I. "Executable Script" (Claim 14) 19

V. Conclusion 20

TABLE OF AUTHORITIES

Page

CASES

Abbott Laboratories v. Novopharm Ltd.,
232 F.3d 1324 (Fed. Cir. 2003).....9, 16

Chime v. PPG Industries, Inc.,
402 F.3d 1371 (Fed. Cir. 2005).....12

Gillette Co. v. Energizer Holdings, Inc.,
405 F.3d 1367 (Fed. Cir. 2005).....9

IPXL Holdings LLC v. Amazon.com, Inc.,
430 F.3d 1377 (Fed. Cir. 2005).....16

Pfizer, Inc. v. Ranbaxy Laboratoriess Ltd.,
457 F.3d 1284 (Fed. Cir. 2006).....15

Phillips v. AWH Corp.,
415 F.3d 1303 (Fed. Cir. 2005).....9, 10, 12

Scimed Life Systems, Inc. v. Advanced Cardiovascular Systems,
242 F.3d 1337 (Fed. Cir. 2001).....10

Springs Window Fashions LP v. Shade-O-Matic Ltd., et al.,
323 F.3d 989 (Fed. Cir. 2003).....10

Vitronics Corp. v. Conceptronics, Inc.,
90 F.3d at 1582 (Fed. Cir. 1996).....9

I. INTRODUCTION

As set forth in detail below, Defendants' proposed constructions are compelled by the relevant intrinsic evidence of the patent at issue, United States Patent No. 5,835,712[1] ("'712 Patent"). By contrast, and as will become clearer from Defendants' motions for summary judgment following construction of these terms by the Court, Plaintiff's constructions are a results-oriented effort to navigate the minefield of invalidity and non-infringement. Plaintiff's constructions on the key terms are not only unsupported by the intrinsic record, they contradict it. The parties' respective constructions are found in Exhibit 1.

II. BACKGROUND

A. The Internet

The '712 Patent generally concerns computer networks, such as the Internet. As shown in Figure 1 of the patent, the Internet (item 10) is a network of many different computers, shown as items 11-14 and 17-19. Like all networks, the Internet allows computers to talk with one another, by exchanging information back and forth. One computer, designated the "client," asks for information. Another computer, the "server," receives that request for information, gathers it, and sends it to back the client over the Internet. (*See* '712 Patent at 6:1-4.)

B. Webpages And The Hypertext Markup Language (HTML)

In the Internet context, the information commonly requested by the client is the ubiquitous "webpage." The client computer runs a computer program called a "Web browser," such as Microsoft's Internet Explorer program. When a computer user requests a webpage, such as by entering the address (or "URL") for a webpage or clicking on a hypertext link, the browser program does a couple of things. First, in ways not relevant to this dispute, it works in conjunction with the Internet to locate the particular server on which the webpage is stored. It then requests that webpage from the server. In simplest terms, the webpage is a computer file, much like a word processing document. However, a webpage is generally written in a

[1] A copy of the '712 Patent is available in Tab A of the Joint Appendix for Claim Construction.

specialized computer language, called Hypertext Markup Language, or "HTML." HTML is the language of the Web, and is understood and used by the browser software on the client computer to display the webpage on a computer monitor. Once the server locates the webpage file requested by the client, the server delivers ("serves") that file back to the client over the Internet.

The client receives the webpage file, and the browser software interprets the HTML language to display the webpage to the user. The HTML language consists of a number of "tags," each of which is a short and simple token that tells the browser software on the client computer how to generate the webpage. The following is a short, but illustrative, example of an HTML computer file that could be sent by a server to a client:

```
<html>
<b>Today is: May 21, 2007</b>
</html>
```

This example includes four HTML tags: <html>, </html>, , and . The first tag, <html>, tells the browser that this file is written in the HTML language. The next tag, , tells the browser to turn on boldface printing. The tag tells the browser to turn off boldface printing. The last tag, </html>, tells the browser that the end of the file has been reached.

When the browser processes this file, the tags are not displayed on the screen. They are only used internally by the browser to format the display. Because the text "`Today is: May 21, 2007`" appears between the "turn boldface on" tag and the "turn boldface off" in the HTML file, the browser program displays that text in boldface on the computer monitor as:

`Today is: May 21, 2007`

C. Dynamic Versus Static Webpages

Webpages can generally be classed as either of two types: static or dynamic. A static webpage is the same each time it is displayed by the browser program on the client computer. A dynamic webpage can (but does not necessarily have to) change. Thus, in the example above, if the browser displays "**`Today is: May 21, 2007`**" every day of the year, it is a static webpage. If date in the webpage changes every day, then it is dynamic.

The ’712 Patent is directed generally to the generation of dynamic webpages. However, as the ’712 Patent itself recognizes, Mr. DuFresne did not innovate the concept of dynamic webpages. The ’712 Patent is instead directed to a very specific way of achieving dynamic webpages. For example, the date in the illustration above could simply be changed manually once per day in the file stored on the server computer. The webpage displayed on the client computer would thus change from day-to-day. While manually updating the file in this way is no more difficult than changing the date in a word processing document, it has obvious drawbacks (some of which are noted in the patent at ’712 Patent at 2:37-44). Thus, long before the filing of the ’712 patent application, myriad technologies existed to generate dynamic Web pages automatically.[2] One such technology discussed in the specification of the ’712 Patent was “Common Gateway Interface” (“CGI”) programs or scripts. The CGI was a feature of servers that allowed the server computer to run a program when it received a request from a client. (’712 Patent at 7:63-64.) In other words, the client computer could either request a webpage from a server computer, tell the server computer to run a program (which the server would run through the CGI), or both. A webpage designer could thus write a computer program—using general-purpose computer language instead of special-purpose Internet language—that, when run by the server, would generate dynamic content. This computer program was called a “CGI script.” This CGI script is not an HTML file, but when run by the server generates a dynamic HTML webpage in real time. (*Id*. at 7:65-8:3; “A CGI script, on the other hand, is executed in real time to output dynamic information which is put into displayed on a Web page. A CGI script is executed when a user activates an HTTP URL that is directed to a file containing a CGI program or script rather than an HTML document.”)

[2] Almost none of these were disclosed to the Patent Office. Defendants will present a subset of this prior art in detail in their summary judgment motions, and later their inequitable conduct case.

The following is an illustration of a CGI script:

```
print ‘<html>’;
print ‘<b>Today is: ’;
print &unixdate(‘today’,’%d’);
print ‘</b></html>’;
exit;
```

In this example, when a client asks the server for the information at an address that points to a CGI script, the server processes that request by executing the script. The CGI script consists of general-purpose computer commands (in green) with HTML codes embedded inside them. The third line shown in the program script “`print &unixdate(‘today’,’%d’)`” causes the server computer to consult its internal calendar to determine the current date. When the server runs this program, the output is exactly the HTML file shown in the example above, i.e.:

```
<html>
<b>Today is: May 21, 2007</b>
</html>
```

This HTML file is sent to the client, where the browser processes it as described above.

The ’712 Patent criticizes CGI scripts. Because CGI scripts are written in general-purpose computer language, they are allegedly “complex,” “difficult to program,” “require customization,” and “lack standardization:”

> CGI scripts, however, are complex and difficult to program. Each script requires customization to implement a particular Web application in a particular way. CGI scripts, therefore, lack standardization and adaptability from one application to another, and content providers running Web servers are often faced with writing multiple CGI scripts to accommodate a variety of HTML forms carried by different Web applications.

(’712 Patent at 8:38-45.)

The ’712 Patent purports to address these shortcomings through a simplified set of “executable tags,” or “tag extensions.” These executable tags are short and simplified HTML-like tokens that can be inserted into webpage files along with, and in some instances embedded into, standard HTML tags. However, unlike standard HTML tags, which are processed by the client computer, these executable tags are processed by the server computer. When executed by

the server, the executable tags are replaced, so that the webpage file delivered to the client computer is entirely composed of standard HTML tags:

> General Framework of the Invention
>
> In a preferred embodiment, the server system of the present invention implements a basic set of components which includes templates, HTML tag extensions, and a structured database system. A template is an HTML form to define contents of a display Web page requested by a client. It is through the use of templates, the server system of the present invention communicates with clients and its own databases. A template includes HTML tags and tag extensions to define and build a Web page. In a preferred embodiment, the extensions are executable codes embedded within conventional HTML tags as arguments. When executed, such a tag extension is replaced with a value to yield a displayable HTML tag. The preferred system of the present invention defines a unique database structure to index data values for retrieval by the server.

(*Id*. at 8:57-9:6.)

Continuing the above example, the following webpage file contains a mixture of both an executable tag (in red) and HTML tags :

```
<html>
<b>Today is: <#TODAY></b>
</html>
```

(*Id*. at 24:38-41.) When the server processes this file, the executable tag `<#TODAY>` is replaced with the current date (`May 21, 2007`), and the output is exactly the HTML file shown in the example above, i.e.:

```
<html>
<b>Today is: May 21, 2007</b>
</html>
```

In this example, the executable tag `<#TODAY>` is positioned between two HTML tags, `<b>` and `</b>`. An executable tag can instead be embedded within a single HTML tag. For example, one very popular HTML tag is the one that creates hypertext links. This tag, in static format, has the following syntax:

```
<A HREF="http://New.com/Dir/Subdir/ITEM-D.html">
```

(*Id*. at 7:24-25.) The information between the quotation marks (`http://New.com/Dir/Subdir/ITEM-D.html`) contains the address of the hyperlink, i.e., the place the file (`ITEM-D.html`) is retrieved from when the hyperlink is selected. In the example above, this address is static, meaning that the user is always directed to the same place.

As set forth in the block quote above describing the "General Framework of the Invention," another feature of the '712 Patent is that the executable tags can be "embedded within conventional HTML tags as arguments." (*Id*. at 9:1-2.) Thus, in the example of the HTML tag that creates a hyperlink, the address can be made dynamic by using one or more executable tags in place of the static address information (here two executable tags, both shown in red):

```
<A HREF="<#URLBASE>form/<#WEB>/calc_xy">
```

(*Id*. at 12:58.) As the patent explains:

> Note that the two tags are simply replaced with the appropriate substitutions. The advantage is that a source path of a URL need not be predefined, and that the code can be used on any system without customization.

(*Id*. at 12:52-54.)

Another feature of the executable tags of the '712 Patent is that, when processed, each is replaced in its entirety by a value. Referring again to the "general framework of the invention" (above), the value is retrieved by the server from a structured database. (*Id*. at 9:4-6; "The preferred system of the present invention defines a unique database structure to index data values for retrieval by the server.") The patent extensively discusses the importance of this structured database in allowing the server to identify the correct value to substitute for the executable tag:

> Creating "databases" is another important aspect of the present invention. Databases themselves are yet another element of an application database. In a preferred embodiment, a database is also defined and maintained by a template/database structure. Database templates provide a platform to create new databases and to open existing databases to copy or modify the stored data values. The preferred syntax for URL is also maintained to retrieve individual database templates. FIG. 12A illustrates a preferred method of defining the database fields. FIG. 12A describes the familiar process for invoking a blank template 141 with a particular URL form as given in 140. The preferred database template form 141 includes the notification field 143. This field lists names of users who will be notified if the database is changed. The text area 146 receives database definitions. The preferred database definitions include a "field name," "type field," and "data field."

(*Id*. at 13:50-67.)

D. The Prosecution History

The application that led to the '712 Patent was filed on May 3, 1996, and contained 65 claims. The Patent Office initially rejected all 65 claims. Plaintiff cancelled many of his claims; the 28 issued claims of the '712 Patent correspond to original claims 1-24 and 41, 42, 55, and 66.

The original claims were all rejected as either anticipated by or obvious over U.S. Patent No. 5,623,656 by Lyons (Tab H;[3] "Lyons") in view of "Inserting Multimedia Objects into HTML3" by Kindel et al. (Tab J; "Kindel").

Lyons discloses a CGI scripting technique, in which standard HTML is inserted into a computer script, and the script is executed at the server computer to generate a dynamic HTML Web page. (Tab B-6 at DuFresne 2124-5.) Kindel discloses executable tags, similar to standard HTML tags, that are included in an HTML file delivered to the client computer. Software at the client computer would both execute the executable tags and process the standard HTML to generate dynamic Web pages. (Tab B-6 at DuFresne 2124.)

In response to these rejections, Plaintiff substantially amended his independent claims, and made extensive arguments, to specify ***where***, ***when***, and ***how*** the "executable tags" within the "hypertext source" are processed to generate dynamic Web pages. The amendments to claim 1 are representative:

> 1. (Amended) A method of deploying client-server applications on a network comprising:
>
> inserting executable tags in a hypertext source to a displayable page; [and]
>
> in response to a request for the displayable page from a client to a server, retrieving the hypertext source and, prior to forwarding the hypertext source to the client, processing the source at the server by executing the tags [in response to a request for the page from a client to a server]; and
>
> forwarding the hypertext source processed at the server to the client.

(Tab B-9 at DuFresne 2137.)

Thus, to distinguish Kindel, and its approach of processing executable tags at the client location to generate dynamic webpages, Plaintiff narrowed the claims to require that the hypertext source must be processed "at the server by executing the tags." Plaintiff's accompanying arguments explained that such server-side processing is more efficient than client-side processing:

> Other advantages of this invention exist as well. Tag execution at the server in the present invention allows page customization for a client before communications with the client

[3] All citations to a "Tab" (alone) refer to a tab in the Joint Appendix for Claim Construction.

> even begins. Execution of tags at the server therefore reduces network traffic since page expansion through tag execution may be done complete within the server without network communications, without user feedback, and without browser interaction. … [T]he use of the present invention reduces network traffic to a minimum since the client is not responsible for executing any tags of the invention, nor is return data required from a client.

(*Id.* at DuFresne 2149.)

Plaintiff's amendments and arguments first sought to distinguish Lyons by reiterating the same general "simplicity" points made in the patent specification to distinguish CGI scripts. According to Plaintiff, his invention involves a simple find-and-replace operation; a single webpage source file contains some simple executable tags, and the server replaces those tags with corresponding values before sending the file to the client computer:

> The present invention as claimed is directed to a method and apparatus for implementing and deploying client-server applications on a network. In the present invention, executable tags are inserted into the source of a displayable page, for example, during page source creation. Then, before a source is sent to a client computer, the server executes any executable tags within the source for that page. As discussed in the specification beginning on page 21 ['712 Patent, at 11:55 *et seq.*] under the heading "HTML Tag Extensions and Instruction Sets", ***the present invention allows dynamic web pages to be presented to users by having the server execute the tags within the source of a page before transmission to a client, in order to substitute the tag with data of various types***. As discussed on page 23, the tags may be replaced or expanded by data and/or instructions which may contain other executable tag extensions which may be further expanded upon.

(*Id.* at DuFresne 2144; emphasis added.) Lyons was also allegedly different because the file that the client ***requested*** from the server (which contained a CGI script written in general-purpose computer language) was different than the HTML file the server ***delivered*** back to the client. Thus, whereas Plaintiff's claims require that the executable tags be executed directly within a "single," "standalone," and "self-sufficient" source file, Lyons' scripts were converted from one source (script) to another (HTML):

> There is no teaching or suggestion in Lyons of an HTML server processing ***a single standalone source*** having executable tags embedded within the script to ***include data into the page***, before that script is sent to a client computer. (Tab B-9 at DuFresne 2146; emphasis added.)

> No server-interpretation of ***executable tags within a single, self-sufficient page*** is taught or suggested by Lyons, or Kindel for that matter. That is, the present invention can ***operate on a single page***… (*Id*. at DuFresne 2147; emphasis added.)

> The present invention provides a way to effectuate dynamic web pages ***while not having to change the source of the web pages at all*** and without requiring user or client interaction. Since in this invention, the server executes the executable tags, the information included into the page

> from tag execution need only change in order to provide customized web pages. ***No hypertext page reprogramming or alteration are needed***…" (*Id.* at DuFresne 2148; emphasis added.)
>
> Each of these independent claims recites that executable tags exist within ***a source*** of a page, and ***that the source*** of the page is processed at the server to execute the executable tags. (*Id.* at DuFresne 2149; emphasis added.)

Moreover, Plaintiff argued, the general-purpose computer programming language scripts in Lyons, and the manner by which they were processed at the server by running the program, was "not at all equivalent or even similar to" the claimed executable tags and the simple find-and-replace technique for swapping tags with corresponding values:

> As described, the script processor [in Lyons], based only on its analysis of the data in the returned data structure and on script selection by the action parameter, can perform four functions referred to as text insertion, conditional text insertion, database access, or conditional script redirection. This is not at all equivalent or even similar to HTML interpretation of the web page at the server.

(*Id.* at DuFresne 2145-6.)

Plaintiff's amendments and arguments persuaded the Examiner to allow the claims asserted in this litigation. (Tab B-10 at DuFresne 2156.)

III. CANONS OF CLAIM CONSTRUCTION

Claim construction involves determining the meaning of the claim to someone having an ordinary level of skill in the art at the time of the invention. *See Phillips v. AWH Corp*., 415 F.3d 1303, 1312-13 (Fed. Cir. 2005) (*en banc*). This task requires the court to place the claim language in its proper technological and temporal context. *Gillette Co. v. Energizer Holdings, Inc*., 405 F.3d 1367, 1370 (Fed. Cir. 2005). The intrinsic evidence, "i.e., the patent itself, including the claims, the specification and, if in evidence, the prosecution history ... is the most significant source of the legally operative meaning of disputed claim language." *Vitronics Corp. v. Conceptronics, Inc*., 90 F.3d at 1582 (Fed. Cir. 1996). The specification "is the single best guide to the meaning of a disputed term." *Phillips*, 415 F.3d at 1315. Where the patentee provides an explicit definition for a claim term in the specification, that definition controls. *Abbott Laboratories v. Novopharm Ltd*., 232 F.3d 1324, 1330 (Fed. Cir. 2003). In the absence of an explicit definition, the specification "can provide guidance as to the meaning of the claims,

thereby dictating the manner in which the claims are to be construed, even if the guidance is not provided in explicit definitional format." *Scimed Life Systems, Inc. v. Advanced Cardiovascular Systems*, 242 F.3d 1337, 1344 (Fed. Cir. 2001).

The prosecution history very often "inform[s] the meaning of the claim language by demonstrating how the inventor understood the invention and whether the inventor limited the invention in the course of prosecution, making the claim construction narrower than it otherwise would be." *Phillips*, 415 F.3d at 1317. If a patentee explicitly states during prosecution that his invention does not cover a particular device, then the claims of the invention must be construed to exclude this disclaimed subject matter. *Springs Window Fashions LP v. Shade-O-Matic Ltd., et al*., 323 F.3d 989, 996 (Fed. Cir. 2003).

Extrinsic evidence, to the extent not at odds with the intrinsic record, and while less significant than the intrinsic record in construing the claims, can in limited circumstances also be useful in claim construction. *Phillips*, 415 F.3d at 1317-18. Technical dictionaries, especially, are "among the many tools that can assist the court in determining the meaning of particular terminology to those of skill in the art of the invention." *Id*. at 1318.

IV. ARGUMENT

A. "Executable Tag" (All Claims)

Plaintiff's Proposed Claim Construction	Defendants' Proposed Claim Construction
Are delimited computer codes capable of being directed to both static and variable values	An HTML-like expression that begins with "<" and ends with ">", wherein the entire expression is replaced with a value upon execution.

1. Defendants' Proposed Construction Is Compelled By The Intrinsic And Extrinsic Evidence

This term presents two issues: (1) what is a "tag?" and (2) what does it mean for the tag to be "executable?" We address these in reverse order.

Although the term "executable tag" is found in all of the claims of the '712 Patent, it occurs only twice in the specification. The first occurrence is in the Summary of the Invention section, in which an "executable tag" is equated with a "tag extension." The remaining portion

of the definition plainly defines "executable" precisely as proposed by Defendants, i.e., replacing the entire executable tag with its corresponding value:

> In a preferred embodiment, the method of deploying client-server applications involves inserting executable tags in an HTML source to a displayable page. These executable tags refer to HTML "tag extensions" defined by the present invention where each tag identifies, from a database, a field name having a value such that ***executing the tags replaces each tag with the corresponding value***. In the preferred embodiment, a Web server, in response to a request for the Web page from a client, processes such a source by executing the tag extensions to expand.

('712 Patent at 3:4-13; emphasis added.)

The second occurrence of the term "executable tag" in the specification provides no additional definition, other than that an "executable tag" is different than an "instruction:"

> The preceding described the general framework of the present invention which includes the client-server communication methods involving templates and executable tags and instructions.

(*Id*. at 13:3-5.)

With regard to the second issue, what it means to be a "tag," the specification uses the synonymous expression "tag extensions" in the same manner set forth in Defendants' construction ("an HTML-like expression that begins with "<" and ends with ">", wherein the entire expression is replaced with a value upon execution"):

> ***The tag extensions of the present invention are a set of HTML-like tags*** that extend the functionality of HTML, simplify the development of Web sites, and permit the implementation of an environment that retains information about previous transactions across the Web pages in a session.
>
> ***The tag extensions simply get replaced by something.*** The key to understanding the tag extensions is, therefore, understanding what gets substituted in place of each tag.
>
> ***The syntax of a tag extension is similar to an HTML tag in that both are enclosed in angle brackets "<>"***; however, the tag extensions are preceded by a pound sign "#". Like HTML tags, the tag extensions may also have one or more modifiers.
>
> Example: <#tag modifier1 modifier2>

(*Id*. at 23:20-38; *see id.* at 3:14-19, 3:52-55, 4:23-25, 4:33-36, 8:67-9:4, 11:58-12:7, 12:5-53, 13:43-47, 15:13-15, 17:11-15, 18:25-29, and 18:44-54.)

Defendants' proposed construction is also compelled by the prosecution history, during which Plaintiff repeatedly emphasized the find-and-replace nature of his alleged invention, where a tag or token is replaced with a corresponding value:

> [T]he present invention allows dynamic web pages to be presented to users by having the server ***execute the tags*** within the source of a page before transmission to a client, ***in order to substitute the tag with data of various types***.

(Tab B-9 at DuFresne 2144; emphasis added.)

The technical dictionary extrinsic evidence from the Free On-Line Dictionary Of Computing ("FOLDOC") likewise confirms the "tag" aspect of Defendants' proposed construction:

> Tag: An SGML, HTML, or XML ***token*** representing the beginning (start tag: "<p ...>") or end (end tag: "</p>") of an element. In normal SGML syntax (and always in XML), ***a tag starts with a "<" and ends with an ">"***.

(Exhibit 2 (Defts' Supp. Appendix for Claim Construction) at Tab T; emphasis added.)

2. Plaintiff's Proposed Construction Flouts The Intrinsic Record

By contrast, Plaintiff's proposed construction for this term ("delimited computer codes capable of being directed to both static and variable values") is not only unsupported by the intrinsic record, it is contrary to it.

Plaintiff's proposed construction attempts now to encompass what he repeatedly said his invention was not. In the specification, and during prosecution, Plaintiff sought to distinguish his alleged invention from prior art techniques including CGI scripts and the scripts in Lyons—which are both collections of computer codes capable of being directed to either or both static and variable values—to create dynamic webpages though server-side processing. Moreover, the Patent Office found that executable scripts and executable tags are different things and Plaintiff acquiesced. (Tab B-10 at DuFresne 2157.) Plaintiff cannot now under the guise of claim construction attempt to recapture that which he disclaimed in order to obtain his patent. *See Phillips*, 415 F.3d at 1316 (specification disclaimer); *Chime v. PPG Industries, Inc*., 402 F.3d 1371, 1384 (Fed. Cir. 2005) (prosecution history disclaimer).

Plaintiff's proposed construction suffers additional defects. First, it does not even attempt to define what it means for the tag to be "executable," thus ignoring half of the relevant inquiry. Second, it improperly defines the singular term "executable tag" as meaning plural "computer codes." This is driven by the way in which Defendants' products work, not by anything in the intrinsic record. Third, it is vague in terms of how the "executable tag" is "delimited" (the term "delimit" is not used anywhere in the '712 Patent), and also as to whether each "executable tag" must be directed to both a static value and a variable value, or whether it is an either-or proposition.

B. "Hypertext Source To A Displayable Page" (All Claims)

Plaintiff's Proposed Claim Construction	Defendants' Proposed Claim Construction
Text with codes that indicate how a page should be displayed by a browser	A single, self-sufficient file consisting of HTML tags and executable tags, where the portion consisting of HTML tags can be displayed as a webpage on the monitor of a client computer using a Web browser.

1. Defendants' Proposed Construction Is Dictated By The Intrinsic Record

There are two principal issues surrounding the construction of this term as well: (1) whether the "hypertext source" must include both HTML tags and executable tags, and (2) whether the source is a "single, self-sufficient file." Both are based on the intrinsic record.

As to the first issue, the independent claims at issue themselves make clear that the "hypertext source" is file containing a mixture of both executable tags and standard HTML tags:

Claim 1: "inserting executable tags in a hypertext source to a displayable page"

Claim 26: "executable tags in a hypertext source to a displayable page"

The executable tags are the ones that are executed by the server, and are either arranged alongside standard HTML tags, or embedded within them. The specification makes clear that the "displayable page" is a file of standard ("pure") HTML tags, of the sort that can be processed by a client computer. This is explained in the Summary Of The Invention as follows:

> The source code defined within the template is then processed by the processor. Such process includes executing the tag extensions in the source to replace each extension with the

> corresponding value so as to configure the page with the remaining hypertext codes. ***The resulting source to the displayable page is, therefore, a pure HTML document which can be interpreted by the client browser***.

('712 Patent at 4:33-39; emphasis added.)

As to the second issue, as noted above, Plaintiff made clear during prosecution, in successfully distinguishing the prior art, that the "hypertext source" was a "single," "standalone," and "self-sufficient" source file that requires no "reprogramming" or "alteration."

2. Plaintiff's Proposed Construction

Plaintiff's construction for this term is, like the last one, overbroad, in that it fails to account for any of the extensive prosecution history on this term.

C. "Processing The Source … By Executing The Tags"/"Processing The Source …, The Process Executing The Executable Tags" (All Claims)

Plaintiff's Proposed Claim Construction	Defendants' Proposed Claim Construction
Executing the executable tags using a program capable of processing static and variable values referred to by the executable tags	Processing the same file into which executable tags were inserted, by replacing each executable tag in the file with its corresponding value.

1. Defendants' Proposed Construction

The principal issue on this claim term is whether the claim requires that the same file be preserved throughout the execution process. That is, Defendants submit that the same source file that contains the executable tags is (1) retrieved, (2) processed to execute the tags, and (3) forwarded to the client after such processing.

Defendants' proposed construction is true to the plain wording of the claims, which first introduce the term "hypertext source" using the indefinite article "a," and then use the definite article "the" to refer back to that same "hypertext source" (claim 1, emphases added):

> inserting executable tags in ***a hypertext source*** to a displayable page;
>
> in response to a request for the displayable page from a client to a server, ***retrieving the hypertext source*** and, prior to forwarding ***the hypertext source*** to the client, processing ***the source*** at the server by executing the tags; and
>
> forwarding ***the hypertext source*** processed at the server to the client.

This means exactly what it says. The invention starts with a hypertext file, "executable tags" are inserted into that same file, that same file is retrieved, that same file is processed (by the server), and that same file is then forwarded to another location (the client).

It was exactly this arrangement and approach that Plaintiff underscored repeatedly throughout prosecution, as evidenced by the passages quoted above emphasizing the simple find-and-replace nature of the claimed invention.

2. Plaintiff's Proposed Construction

There are three defects in Plaintiff's proposed construction ("executing the executable tags using a program capable of processing static and variable values referred to by the executable tags"). First, Plaintiff's proposed construction ignores both that the plain language of the claims—in which the same file is preserved throughout the claimed process—as well as the prosecution history underscoring the importance of this requirement. (*See* Tab B-9 at DuFresne 2146-9; plaintiff discusses importance of preserving the same file (discussed above); *see also* Tab B-10 at DuFresne 2157; examiner rejects claim 34 (also discussed above).)

Second, the claim clearly states that processing the source requires execution of the "executable tags" themselves, not a program that processes values referred to by the tags. As a result of the claimed execution of the tags themselves, the tags are replaced with their corresponding values: "executing the tags replaces each tag with the corresponding value." ('712 Patent, at 3:9-10.) Thus, the hypertext source initially has executable tags, and then after processing has values in place of those tags. Plaintiff's proposed construction gets this backwards, in proposing that the "static and variable values referred to by the executable tags" are executed, and not the executable tags themselves, as the claims plainly require.

Third, Plaintiff improperly tries to rewrite the claim to include a "program." None of the patent claims—independent or dependent—require a "program." There is no basis in the intrinsic record to insert "program" into any of the claims. Plaintiff is improperly attempting to use claim construction fix a fatal error in claim 26. *Pfizer, Inc. v. Ranbaxy Labs. Ltd.*, 457 F.3d

1284, 1292 (Fed. Cir. 2006) ("[W]e 'should not rewrite a claim to preserve its validity.'"). As will be explained in detail in Defendants' motion for summary judgment, claim 26 is a mixed apparatus and method claim. The claim is principally directed to a system, but only when the system is performing the "processing" and "executing" steps required by the claim. The Federal Circuit has made clear that mixed method and apparatus claims of this sort are invalid as a matter of law. *IPXL Holdings LLC v. Amazon.com, Inc.*, 430 F.3d 1377, 1384 (Fed. Cir. 2005) ("Because claim 2 recites both a system and the method for using that system, it does not apprise a person of ordinary skill in the art of its scope, and it is invalid under section 112, paragraph 2.")

D. "Server" (All Claims)

Plaintiff's Proposed Claim Construction	Defendants' Proposed Claim Construction
A computer or computer software that receives requests for information from a client	A computer that receives requests for information from a client.

The parties' constructions for "server" are close: both agree that a server is a computer that receives requests for information from a client. However, Plaintiff's construction includes the alternative possibility that this could be carried out by "computer software" ***only***.

Defendants of course do not dispute that the computer in their proposed construction would be running software. Indeed, the specification makes this clear when it defines the term: "A server is a system running a program that manages access to stored information." ('712 Patent at 1:47-49.) Where, as here, an inventor has expressly defined a claim term, he acts as his own lexicographer and that definition controls. *Abbott Labs.*, 232 F.3d at 1330. ('712 Patent at 6:3-9 ("The computers on the Internet are largely classified as either servers 13, 14, 18, or clients 11, 12, 17, 19"); *id.* at Figure 1; *id*. at 6:23-26 ("Web server 20 refers to a machine on the Internet which runs a program…").)

E. "Client" (All Claims)

Plaintiff's Proposed Claim Construction	Defendants' Proposed Claim Construction
A computer or computer software that requests information from a server	A computer that requests information from a server.

The same issues apply to the consideration of this term, which is in some sense the flip-side of the "server." Like "server," this term is expressly defined in the specification to mean a computer (i.e., a system): "a client is a system that makes a request for certain information stored and controlled by the server." ('712 Patent at 1:49-50; *see also id.* at 2:15-18 ("Upon receiving a request for a Web page, a server typically returns an HTML document which is decoded and displayed on a Web browser running on the client's system."); *id*. at 6:3-9 ("The computers on the Internet are largely classified as either servers 13, 14, 18, or clients 11, 12, 17, 19.) Moreover, while prosecuting the '712 Patent, Plaintiff repeatedly referred to a "client" as a "client computer." (*See, e.g.*, Tab B-9 at DuFresne 2144 ("before a source is sent to a client computer"); *id.* at DuFresne 2145 ("user at a client computer," "sent to a client computer," "received from the client computer"); *id.* at DuFresne 2146 ("sent to a client computer," "client computer browser"); and *id.* at DuFresne 2148 ("in every client computer").)

F. "Hypertext Markup Codes"/"Markup Codes" (Claims 9 and 26)

Plaintiff's Proposed Claim Construction	Defendants' Proposed Claim Construction
In a hypertext language, the computer codes within delimiters that indicate how a document is interpreted by a browser	An HTML expression that begins with "<" and ends with ">" and that is used by a Web browser on the client computer to display the webpage on the monitor of the client computer.

Defendants' construction is fully consistent with the intrinsic and extrinsic record. The sole discussion of "markup" in the specification is in the context of standard HTML tags: "The information to be transferred on the WWW is drafted with the HTML (Hypertext Markup Language). The HTML documents are made up of standard text formatted with a special set of codes which indicate how the document should be displayed." ('712 Patent at 2:10-15.) The patent further makes clear that an HTML tag begins with a "<" symbol and ends with a ">" symbol: "The most basic element in an HTML document is a tag, which is enclosed by the angle brackets, '<' and '>'." (*Id*. at 6:43-45.) Defendants' construction also fully comports with the industry standard definitions. (*See generally* the above-quoted technical dictionary definition of "tag" and the definition of "HTML:" "A hypertext document format used on the World-Wide

Web. HTML is built on top of SGML. 'Tags' are embedded in the text. A tag consists of a '<', a 'directive' (in lower case), zero or more parameters and a '>'." (Exhibit 2 at Tab U.))[4]

G. "Embedded Within The [Markup] Codes" (Claims 9 and 26)

Plaintiff's Proposed Claim Construction	Defendants' Proposed Claim Construction
Between	Placed between the "<" and ">" symbols bounding respective markup codes.

Given the ambiguity in Plaintiff's proposed construction, it is unclear whether the parties disagree on this claim term. Defendants' construction makes clear that when, as required in both claim 9 and claim 26, an "executable tag" is "embedded within" a markup code (i.e., a HTML tag), the executable tag is placed between the "<" and ">" symbols bounding the HTML tag. Defendants' construction is true to the usage of the term "embedded" in the "General Framework of the Invention" section of the patent, which states that the executable tags can be "embedded within conventional HTML tags as arguments." ('712 Patent, at 9:1-2.) Thus, as noted above, executable tags can be inserted directly into the body of an HTML tag (here two executable tags, both shown in red):

```
<A HREF="<#URLBASE>form/<#WEB>/calc_xy">
```

(*Id*. at 12:58.) As the patent explains:

> Note that the two tags are simply replaced with the appropriate substitutions. The advantage is that a source path of a URL need not be predefined, and that the code can be used on any system without customization.

(*Id*. at 12:52-54.)

H. "Database" (Claim 26)

Plaintiff's Proposed Claim Construction	Defendants' Proposed Claim Construction
A collection of data for retrieval	A structured set of data.

[4] Defendants recognize that claim 9 recites "markup codes," whereas claim 26 uses the term "hypertext markup codes." While principles of claim differentiation would generally suggest that there is a difference between these two terms, with the former being broader than the latter, the patent specification does not meaningfully distinguish between the two. Moreover, Plaintiff agrees that the two terms should be given the same construction. Since it would be improper to read the word "hypertext" out of claim 26, Defendants submit that the narrow construction is the appropriate one to choose.

This term is only present in claim 26. The single issue that divides the parties on this term is whether the data in the database must be structured. Defendants submit that it does.

The specification of the '712 Patent explains that the "preferred system of the present invention defines a unique database structure to index data values for retrieval by the server." ('712 Patent, at 9:4-6.) This structured database is identified as "another important aspect of the present invention." (*Id*. at 13:50-58.) The database structure is mentioned in the "General Framework of the Invention" section, particularly with reference to Figure 7 (*id*. at 9:19-47), the section entitled "Templates/Database Structure" (*id*. at 9:48-11:52, particularly with reference to Figure 9B), and the section entitled "Databases" (*id*. at 13:48-16:14). The entire description is consistent with a unique structured set of data that is accessible in a client-server environment.

So too is Defendants' proposed construction supported by the extrinsic evidence of the FOLDOC computer dictionary: "Database: One or more large structured sets of persistent data, usually associated with software to update and query the data. A simple database might be a single file containing many records, each of which contains the same set of fields where each field is a certain fixed width." (Exhibit 2 at Tab V.)

By contrast, Plaintiff's proposed construction is so overbroad as to essentially read the term "database" out of the claim. "A collection of data for retrieval" could simply refer to any information, regardless of how stored, maintained, or retrieved. That notion is already present in the definition of an executable tag, and so the ***additional*** requirement of a "database" would, on Plaintiff's proposed construction, be meaningless.

I. "Executable Script" (Claim 14)

Plaintiff's Proposed Claim Construction	Defendants' Proposed Claim Construction
collections of computerized instructions which when run perform predefined tasks such as mathematical computations or complex data arrangements	A collection of executable tags and instructions, where the collection is executed in sequence to perform a prescribed task.

The term "script" occurs only in asserted claim 14. Defendants' proposed construction simply adopts the express definition in the specification, with a modest amount of clean up for

clarification purposes: "a 'script' is a collection of extension tags and instructions sets executed in sequence to perform a prescribed task." (*Id*. at 16:18-20.)

Plaintiff's proposed construction is defective in two respects. First, it does not require that the script has "executable tags," even though claim 13 (from which claim 14 depends) expressly includes this requirement: "the source includes executable scripts each including a series of other executable tags to perform predefined tasks." Second, it contains the vague "illustrative" language of "such as mathematical computations or complex data arrangements," two requirements that are not imposed on scripts by anything in the intrinsic record.

V. CONCLUSION

For the foregoing reasons, Defendants request that the claims be construed as proposed by Defendant in the attached Exhibit 1.

Dated: May 14, 2007

/s/ Frank E. Scherkenbach
Frank E. Scherkenbach (BBO #653819)
Kurt L. Glitzenstein (BBO #565312)
FISH & RICHARDSON P.C.
225 Franklin Street
Boston, MA 02110-2804
TEL: (617) 542-5070

Attorneys for Defendants
MICROSOFT CORPORATION, ADOBE SYSTEMS INCORPORATED and MACROMEDIA, INCORPORATED

Dated: May 14, 2007

/s/ Robert S. Frank, Jr.
Robert S. Frank, Jr. (BBO #177240)
CHOATE, HALL & STEWART LLP
Two International Place
Boston MA 02110
TEL.: (617) 248-5000

Attorneys for Defendants
ADOBE SYSTEMS INCORPORATED and MACROMEDIA, INCORPORATED

Certificate of Service

I hereby certify that Defendants' Opening Claim Construction Brief, along with its accompanying Exhibits 1 and 2, was filed through the ECF system will be sent electronically to the registered participants as identified on the Notice of Electronic Filing (NEF). A paper copy of this filing will be sent to those indicated as non-registered participants on May 14, 2006.

/s Jason W. Wolff
Jason W. Wolff

UNITED STATES DISTRICT COURT
DISTRICT OF MASSACHUSETTS

FRED B. DUFRESNE, Plaintiff, v. MICROSOFT CORPORATION, ADOBE SYSTEMS INCORPORATED and MACROMEDIA, INCORPORATED, Defendants.	Civil Action No. 02-11778 WGY

INDEX OF EXHIBITS

Exhibit 1: Table of Proposed Claim Constructions

Exhibit 2: Defendants' Supp. Appendix for Claim Construction